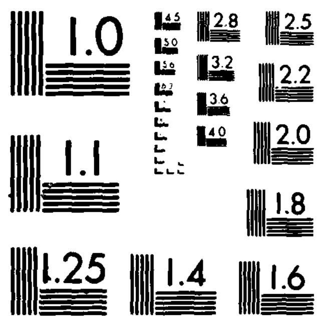
AD-A111 705 FORD AEROSPACE AND COMMUNICATIONS CORP PALO ALTO CA W--ETC F/G 9/2
KSOS SYSTEM SPECIFICATION (TYPE A) (KERNELIZED SECURE OPERATING--ETC(U)
NOV 80 MDA903-77-C-0333
UNCLASSIFIED WDL-TR7808-REV-2 NL

END
DATE
FILMED
4-82
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

WDL-TR7808
Revision 2
November 1980

SECURE MINICOMPUTER OPERATING SYSTEM (KSOS)

SYSTEM SPECIFICATION (TYPE A)

ADA111705

Department of Defense Kernelized Secure Operating System

Contract MDA 903-77-C-0333
CRDL No. 0002AB

Prepared for:

Defense Supply Service-Washington
Room 1D245, The Pentagon
Washington, D.C. 20310

Approved for public release; distribution unlimited.

Ford Aerospace &
Communications Corporation
Western Development
Laboratories Division

3939 Fabian Way
Palo Alto, California 94303

NOTICE

The Department of Defense Kernelized Secure Operating System (KSOS) is being produced under contract to the U.S. Government. KSOS is intended to be compatible with the Western Electric Company's UNIX*tm Operating System (a proprietary product). KSOS is not part of the UNIX license software and use of KSOS is independent of any UNIX license agreement. Use of KSOS does not authorize use of UNIX in the absence of an appropriate licensing agreement.

This document is a minor revision of an earlier version dated July 1978. That and other previous editions are obsolete and should not be used.

UNIX and PWB/UNIX are trade/service marks of the Bell System.

DEC and PDP are registered trademarks of the Digital Equipment Corporation, Maynard MA.

CONTENTS

KSOS System Specification (Type A)

Ford Aerospace and Communications Corporation

Western Devlopment Laboratories Division

1. SCOPE

1.1 Identification

This specification establishes the performance, design, development and test requirements for the Kernelized Secure Operating System (referred to as "KSOS"). KSOS provides a provably secure, resource-sharing operating system compatible with the standard user environment provided by UNIX*tm, as described by [Thompson 75] and [Ritchie 74].

1.2 Document Outline

This specification is organized as follows. Section 2 contains the referenced document citations. Section 3 contains the design requirements for KSOS. Section 4 contains the quality assurance provisions for KSOS. Sections 5 and 6 are not applicable to this specification and are null.

(The language used throughout this specification attempts to conform to the guidelines of Section 3.2.3 of MIL-STD-490. In particular, the word "shall" means that the specification expresses a provision that is binding. The words "should" and "may" mean that the specification expresses a provision which is non-mandatory. The word "will" is used to express a declaration of purpose on the part of the Government.)

2. APPLICABLE DOCUMENTS

The following documents, of exact issue shown, form a part of this specification to the extent specified herein. In the event of a conflict between the referenced documents and the contents of this specification, this specification shall be considered a superseding requirement. In the text references to these documents are in the form [Name date], e.g. [Biba 75].

2.1 Government Documents

2.1.1 Directives, Manuals and Standards

a. DoD 5200.1-R Information Security Program Regulation

b. DoD 5200.28 Security Requirements for Data Processing (ADP)

c. DoD 5200.28-M ADP Security Manual

d. MIL-STD-483 Configuration Management

e. MIL-STD-490 Specification Practices

f. MIL-STD-1521A Technical Reviews and Audits

2.1.2 Reports

a. [Bell and LaPadula 73] Bell, D.E. and LaPadula, L.J., "Secure Computer Systems", ESD-TR-73-278, Volume I-III, MITRE Corporation, Bedford, MA (November 1973 - June 1974).

b. [Biba 75] Biba, K.J., "Integrity Considerations for Secure Computer Systems", MTR-3153, MITRE Corporation, Bedford, MA (June 1975).

c. [Walter et al. 74] Walter, K.G. et al., "Primitive Models for Computer Security", ESD-TR-74-117, Case Western Reserve University, Cleveland, OH (January 1974).

2.2 Non-Government Documents

a. [BBN 75] "Interface Message Processor, Specification for the Interconnection of a Host and an IMP", Report 1822, Bolt Beranek and Newman, Inc., Cambridge, MA (December 1975).

b. [Cerf 77] Cerf, V., "Specification of Internet Transmission Control Program", Version 3 (draft), December 1977. (Available from DARPA, 1400 Wilson Blvd., Arlington, VA 22209.)

c. [Holmgren et al. 77] Holmgren, S.F., Healy, D.C., Jones, P.B. and Kasprzycki, E., "Illinois Inter-Process Communication Facility for UNIX", CAC Technical Memorandum 84, CCTC-WAD Document 7507, Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, IL (April 1977).

d. [Lampson 73] Lampson, B., "A Note on the Confinement Problem", CACM, Volume 16, Number 10, pp 613 - 615 (October 73).

e. [Lipner 75] Lipner, S.B., "A Comment on the Confinement Problem", Proc. Fifth Symposium on Operating Systems Principles, ACM SIGOPS Review, Volume 9, Number 5, pp 192 - 196 (19-21 November 1975).

f. [Nemeth et al. 77] Nemeth, A.G., Sunshine, C., Zucker, S. and Tepper, S., "Progress Towards a DeFacto DoD UNIX Inter-Process Communication Standard", Working Paper, (May 1977). (Available from the authors.)

g. [Parnas 72] Parnas, D.L., "A Technique for Software Module Specification with Examples", CACM, Volume 15, Number 5, pp 330 - 336 (May 1972).

h. [Ritchie 74] Ritchie, D.M. and Thompson, K., "The UNIX Timesharing System", CACM, Volume 17, Number 5, pp 365 - 375 (May 1974).

i. [Sunshine 77] Sunshine, C., "Interprocess Communication Extensions for the UNIX Operating System: I Design Considerations", R-2064/1-AF, RAND Corporation, Santa Monica, CA (June 1977).

j. [Zucker 77] Zucker, S., "Interprocess Communication Extensions for the UNIX Operating System: II Implementation", R-2064/2-AF, RAND Corporation, Santa Monica, CA (June 1977).

2.3 Other References Not Part of This Specification

a. [Thompson 75] Thompson, K. and Ritchie, D.M., "UNIX Programmer's Manual", Sixth Edition, Western Electric Corporation, Greensboro, NC (May 1975).

b. [UNIX 75] "UNIX Program Listings", Version 6.0, Western Electric Corporation, Greensboro, NC (May 1975).

3. REQUIREMENTS

3.1 System Definition

KSOS is intended to be an operating system for minicomputers. The KSOS system call interface shall be compatible with the existing UNIX interface. KSOS shall also include the maintenance and support programs necessary for continued, reliable operation of the system.

3.1.1 General Description

KSOS consists of three (3) Computer Program Configuration Items (CPCI's):

3.1.1.1 Security Kernel - CPCI Number 1

The Security Kernel is the fundamental component of the operating system. The Security Kernel provides a primitive, virtual computational environment which can be proven to faithfully implement DoD security policy and UNIX discretionary security policy.

3.1.1.2 UNIX Emulator - CPCI Number 2

The UNIX Emulator transforms the primitive computational environment provided by the Security Kernel into an environment similar to the existing UNIX user environment.

3.1.1.3 Non-Kernel Security Related Software - CPCI Number 3

The Non-Kernel Security Related Software is a collection of autonomous subsystems which provide essential services to the system. This CPCI is further subdivided into two classes:

a. trusted Non-Kernel Security Related Software which performs functions that are critical to the system's security and may be afforded more privileges than normal user programs

b. untrusted Non-Kernel Security Related Software which performs functions that do not require any extra privileges, and which cannot violate the system's security rules

The services provided by the Non-Kernel Security Related Software include, but are not limited to, the following:

a. Secure User Services, those services invoked by users which must have a trusted path to the service, such as system login.

b. System Operation Services, those functions essential to the operation of a KSOS system, such as the Network Daemon.

c. System Maintenance Services, those functions needed for continued operation and maintenance of a KSOS system, such as dump and restore of file systems.

d. System Administrator Services, those functions needed to support the administrative operation of the system, such as the adding and deleting of users.

These functional components shall implement the reference monitor concept [Bell and LaPadula 73]. The reference monitor (via a combination of hardware and software) mediates every access attempt. In KSOS security-relevant decisions shall be localized in the Security Kernel and the trusted processes of the Non-Kernel Security Related Software. The size of the Security Kernel and the individual trusted process of the Non-Kernel Security-Related Software shall be minimized to allow them to be rigorously specified and eventually verified. KSOS shall also be designed to offer good performance. The performance goals are presented in Section 3.2.1.

KSOS shall support the mandatory DoD security policy of DoD Directive 5200.1-R that is embodied in a Government approved mathematical model. Proofs of the system's security properties shall be in terms of this model. KSOS shall provide a minimum of eight (8) hierarchical security classifications, and a minimum of thirty-two (32) mutually independent security categories. The security levels shall be such that

UNCLASSIFIED < CONFIDENTIAL < SECRET < TOP SECRET

Where "<" is defined in accordance with the requirements of DoD 5200.1-R. One security category shall be reserved for read protection of system data bases and programs. This category shall be called the "system" category. The Government may elect to predefine the specific meaning of a given set of KSOS security categories to facilitate information interchange between systems. KSOS shall provide for Kernel-enforced integrity. Integrity is defined as the formal mathematical dual of security [Biba 75]. At least four (4) hierarchical integrity classifications shall be provided in KSOS. The integrity classifications include at least the following three classifications

USER < OPERATOR < ADMINISTRATOR

Although there is no immediate requirement for integrity categories, the development contractor should include provisions to ease their later inclusion. For the remainder of this specification, the term "security level" means the combination of a security classification and a set of security categories (which may be null). The term "integrity level" means the combination of an integrity classification, and a (presently always null) set of integrity categories. The term "level" means the security and integrity level, that is, the combination of a security classification, a set of security categories (which may be null), an integrity classification, and a (presently always null) set of integrity categories.

KSOS shall also support a discretionary access policy similar to that presently found in UNIX [Ritchie 74] and [Thompson 75].

KSOS should provide facilities to allow ease of operation in a multilevel security environment. Such facilities include, but are not limited to, the following:

a. controlled changing of the classification of files

b. manipulation of the user's current security and integrity levels

c. audit trails in accordance with DoD Directive 5200.28-M

3.1.2 Missions

KSOS shall provide a general-purpose computational environment compatible with the existing UNIX user environment. Typical uses of this environment are situations in which a group of users at different security levels (different clearances, different need-to-know categories or combinations of the two) are to be provided with a shared computational resource. Examples of such situations include secure network front ends, multi-level-secure research facilities, and office automation systems. The KSOS Security Kernel shall be designed to facilitate its application to roles other than providing a UNIX-like environment. Typical of such environments are single-purpose, multi-level-secure systems, such as highly specialized message processing systems.

3.1.3 Threats

KSOS shall be designed and implemented to allow unprivileged execution of an arbitrary program with no security violations. The creator of user-mode programs should be assumed to have access to all the code and documentation for KSOS. Thus, these user-mode programs may attempt to exploit any systematic errors of omission or commission in the system. KSOS shall be designed and implemented to protect against both direct assaults and so-called "confinement channels" ([Lampson 73] and [Lipner 75]). For each identified confinement channel, an estimate shall be made of the channel bandwidth and the performance degradation (if any) required to reduce this bandwidth. The Government will provide direction on a case by case basis on the method of controlling the confinement channels.

3.1.4 System Diagrams

The relationship of the three KSOS CPCI's is shown in Figure 1.

```
              .____________________________________________.
              | User Programs| untrusted    .              |
USER MODE     | (may include | NKSR         .              |
              | Kernel calls)|              .              |
              |______________|______________...............|
SUPERVISOR    |                             |              |
MODE          |        UNIX EMULATOR        | trusted NKSR |
              |_____________________________|______________|
KERNEL MODE   |          SECURITY KERNEL                   |
              |____________________________________________|
```

(NKSR: Non-Kernel Security Related Software)

Figure 1. KSOS CPCI Relationships

Interactions across the execution domain boundaries (e.g., user-mode programs interacting with the UNIX Emulator or the UNIX Emulator interacting with the Security Kernel) shall be via a hardware assisted domain transition. (This hardware assisted transition has been called a "trap" in some computer systems.)

3.1.5 Interface Definitions

The KSOS system call interface shall be closely compatible with the existing UNIX system call interface. Deviations from the existing interface shall require individual justification. A design goal is that existing UNIX user-mode software should run correctly under KSOS, except where such software violates the multi-level security model. This design goal is intended to mean that the binary images of existing UNIX software will run under KSOS without modification, providing that the software does not violate the security model.

KSOS shall support a hierarchical file system like that of existing UNIX. Directories themselves may have a security and integrity level that is independent of the security and integrity levels of the files under them. Users of such multi-level directories must take care that the names of their files are at or below the level of the directories in which the names are found. The situation is analogous to the need for UNCLASSIFIED titles of classified reports.

Selected user-mode programs may interact directly with the Security Kernel, rather than going through the UNIX Emulator. This ability may be administratively restricted (similarly to the way in which other privileges are [illegible]ed to processes) to prevent such programs from affecting other users. [illegible] potential restrictions may be removed if the development contractor can demonstrate that no security flaws result from their removal.

3.1.5.1 Basic KSOS System Calls

This section defines the user interface to the KSOS operating system. The KSOS operating system shall emulate the existing UNIX user interface within the constraints of the multi-level security model. The accepted definition of the existing UNIX operating system interface is [Thompson 75]. In general, those undocumented features which may exist in a particular version of UNIX need not be supported in KSOS. Those functions where the development contractor may exercise design discretion are separately identified.

The concept of a process family is used in several of the calls. A process family consists of the processes forked by a particular user and the descendents of these processes. The development contractor may alter the notion of a process family to improve system effectiveness, providing there is no compromise of security or compatibility. Government approval of the resulting definition of a process family is required.

In all the system calls, violations of the security policy will result in the call failing, and the return of a descriptive error code.

3.1.5.1.1 break

In KSOS the "break" may set the program break address to the nearest integral KSOS storage unit boundary greater than the request. All other effects shall be the same as the UNIX call.

3.1.5.1.2 chdir

The KSOS "chdir" call shall require read access to the desired directory and read or search access to each directory in the intended path. The effects of the KSOS "chdir" call shall be the same as the existing UNIX call.

3.1.5.1.3 chown

The development contractor may replace the "chown" call with another mechanism of comparable functionality, subject to Government approval. The "chown" function should be controlled similarly to changing the classification of a file.

3.1.5.1.4 chmod

The "chmod" call shall require that the calling process be the current owner of the file or directory. The effects of the KSOS "chmod" call shall be the same as the UNIX call.

3.1.5.1.5 close

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.6 creat

The "creat" call shall require the calling process to have write access in the parent directory, and read or search access along the entire path to the parent directory. In KSOS "creat" shall create a file at user's current security and integrity level. The user's current level must allow the updating (reading and writing) of the parent directory. If the user's current level does not allow updating of the parent directory, the "creat" call will fail, and return a descriptive error code. The development contractor may suggest alternative (trusted) mechanisms to allow the creation of files under directories whose updating would be prohibited by the security model. Otherwise the effect of the "creat" call shall be the same as the existing UNIX call.

3.1.5.1.7 csw

The "csw" (read the console switches) call shall not be supported in KSOS.

3.1.5.1.8 dup

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.9 exec

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.10 exit

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.11 fork

No change shall be made to the specification of the existing UNIX call. The security and integrity level of the child (forked) process shall be that of the parent.

3.1.5.1.12 fstat

The status block returned by "fstat" and "stat" shall omit shared information, such as times of last access and global link counts, that could be used for storage channels. The remaining information and its format shall be the same as the existing UNIX call.

3.1.5.1.13 getgid

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.14 getpid

The process identification of the process shall be returned to the calling process. No information shall be returned about processes in other families.

3.1.5.1.15 getuid

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.16 gtty

Read access to the device (file) shall be required. Otherwise, the "gtty" call shall function as it does in standard UNIX.

3.1.5.1.17 indir

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.18 kill

In KSOS the levels of the sending and receiving processes must allow communication (i.e. that the sender can write to the recipient) or else the "kill" call shall fail. Otherwise there shall be no change from the effects in UNIX.

3.1.5.1.19 link

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.20 mknod

The call "mknod" shall require the calling process to have write access to the parent directory (i.e. the directory in which "mknod" would create the new entry), and read or search access along the entire path to the parent directory. The calling process shall not be required to be a privileged user to create directories. The creation of special files (devices) shall be restricted to processes running at (or above) the ADMINISTRATOR integrity level. Special files may be indicated by a different mechanism than in UNIX. The security and integrity level of the directory created by "mknod" shall be the user's current level. The user's current level must allow updating (reading and rewriting) of the parent directory or "mknod" will fail and return a descriptive error code. The development contractor may suggest alternative (trusted) mechanisms that allow the creation of directories under parent directories whose updating would be prohibited by the security model. All other effects of "mknod" shall be the same as the existing UNIX call.

3.1.5.1.21 mount

The "mount" system call shall be the same as in UNIX. However, its use shall be restricted to processes running at (or above) the OPERATOR integrity level. Normally, the "mount" call would be issued only by a privileged process which would perform additional checks prior to attempting the "mount". The development contractor may suggest alternative mechanisms which would subsume the "mount" call. Should the Government approve these alternate mechanisms, the "mount" call may be deleted from KSOS.

3.1.5.1.22 nice

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.23 open

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.24 pipe

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.25 profil

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.26 ptrace

The effects of "ptrace" in KSOS shall be limited to the current process family. Otherwise, the effects shall be as in UNIX.

3.1.5.1.27 read

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.28 seek

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.29 setgid

The "setgid" call shall only be allowed to revert the effective group identification back to the real group identification. The changing of real group identifications, and the setting of the effective group identification to other than the real group identification shall be provided by trusted, Non-Kernel Security Related Software.

3.1.5.1.30 setuid

The "setuid" call shall only be allowed to revert the effective user identification back to the real user identification. The changing of real user identifications, and the setting of the effective user identification to other than the real user identification shall be provided by trusted, Non-Kernel Security-Related Software.

3.1.5.1.31 signal

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.32 sleep

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.33 stat

See the comments on "fstat". The user shall be required to have read or write access to the file and read or search access along the entire path to it.

3.1.5.1.34 stime

This call may be deleted if equivalent functionality is provided as part of the pre-operational or initialization phases of KSOS operations.

3.1.5.1.35 stty

The "stty" call shall require the user to be the current owner of the affected device. The development contractor may provide additional device special function capability which subsumes or replaces the "stty" call. Government approval of all such proposed changes is required.

3.1.5.1.36 sync

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.37 time

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.38 times

Execution times for both user-mode and UNIX Emulator execution shall be returned. Security Kernel execution time shall be unobtainable. Otherwise the "times" call shall operate in the same manner as the existing UNIX call.

3.1.5.1.39 umount

See the comments on the "mount" call.

3.1.5.1.40 unlink

The calling process shall be required to have write access to the parent directory, read or search access along the entire path to the parent directory and write access to the file being removed. Otherwise the call shall perform in the same fashion as the existing UNIX call.

3.1.5.1.41 wait

In KSOS the inheritor of orphaned children may be a different process than process number 1 as in UNIX. Otherwise the call shall perform in the same fashion as the existing UNIX call.

3.1.5.1.42 write

No change shall be made to the specification of the existing UNIX call.

3.1.5.1.43 abort

No change shall be made to the specification of the existing UNIX call.

3.1.5.2 Calls Not Present in Version 6.0 UNIX

3.1.5.2.1 gprocs

The "gprocs" call is not implemented in standard UNIX. Its purpose shall be to return (a portion of) the process table, showing the state of the active processes. Only processes whose levels are such that the caller can observe them shall be represented in the returned table. The format of the returned table shall be that of the existing UNIX process table. The call shall also return the number of processes represented in the returned table.

3.1.5.2.2 getal

The "getal" call is not currently implemented in standard UNIX. "getal" shall return the current level (security classification, security category set, integrity classification, and integrity category set) of the calling process.

3.1.5.2.3 sfork

The "sfork" call is not currently implemented in standard UNIX. "sfork" shall be equivalent to "fork" except that the child process shall be immune to terminal-generated interrupt and quit signals. The child process shall still be susceptible to interrupt and quit signals sent from other processes whose security and integrity levels allow the sending of such signals to the child process.

3.1.5.2.4 eofp

The "eofp" call is not currently implemented in standard UNIX. "eofp" shall cause the data stream in a pipe to have an end of file "mark" inserted into it. Attempts by the reader to read past the end of file shall result in an error return from the "read" call.

3.1.5.3 Additional System Calls and Modifications

Subject to Government approval, the development contractor may provide additional system calls and/or modifications to existing UNIX calls. Such additions and modifications shall be acceptable only if they provide essential features unobtainable through other calls or offer significant cost or performance advantages over existing calls.

3.1.5.4 Inter-process Communication

KSOS shall include operating system calls for inter-process communication (in addition to the UNIX kill/signal mechanisms). These calls shall reflect the concepts in [Zucker 77], [Sunshine 77], [Holmgren et al 77], and [Nemeth 77]. The exact form of the calls is left to the discretion of the development contractor, subject to Government approval. The KSOS inter-process communication mechanism shall be subject to the same type of security constraints as any other access attempt.

3.1.5.5 Computer Network Interface

KSOS shall support an interface to a computer network similar to the ARPANET. The network protocol shall be the TCP [Cerf 77]. The network interface shall be in accordance with [BBN 75], commonly referred to as an "1822 Interface". Both local and Very Distant Host (VDH) interfaces shall be supported by KSOS. To the user, the network should appear as a standard device, if possible. That is, the KSOS "open", "close", "read", and "write" calls should be employed for user-mode network use. A mechanism shall be provided for specifying the TCP port and other information needed by the TCP software to establish the requested communication.

3.1.5.6 Device Files

UNIX disk device files (e.g. /dev/hp??) shall be supported in KSOS. Whenever KSOS file systems are physically present in the device, the security level of the device file shall raised to at least the highest security level in the file system. In addition the integrity level of the device file shall raised to at least the OPERATOR level. Device files that are not being used for KSOS file systems may be permitted only if it can be shown not to compromise security.

Each device shall have a maximum security level that is specified at system generation. This maximum security level will be determined by the System Security Officer (or the System Administrator) and should consider the the physical location of the device, the clearances of personnel having access to the device, and the security level of any communications lines to the device. This maximum level shall not be altered while the system is in normal operation. KSOS shall allow an appropriately authorized user to set the security level of a device to an arbitrary level less than or equal to its maximum level.

The UNIX memory device files, /dev/mem and /dev/kmem, shall not be supported in KSOS.

3.1.5.7 Auditing Interface

KSOS shall contain the necessary internal mechanisms to allow for auditing of access attempts, both successful and unsuccessful, in accordance with Section V of DoD Directive 5200.28-M. The interface for such mechanisms is left to the discretion of the development contractor, subject to Government approval.

3.1.6 Government Furnished Property List

The KSOS Security Kernel and UNIX Emulator shall be a replacement for the existing UNIX operating system. The KSOS Non-Kernel Security-Related Software shall replace and extend the "trusted" UNIX software, such as /bin/login etc. A KSOS installation requiring a full UNIX-like environment shall be able to incorporate the presently (Government) distributed UNIX user-mode system software (compilers, utilities, editors, etc.) except where such software violates the security model or does not function correctly because of changes to the UNIX call interface in KSOS. In such cases, replacement software modules of comparable functionality or modified versions of UNIX modules shall be distributed with the KSOS operating system. This replacement software shall be produced by the development contractor or others as designated by the Government.

KSOS shall be fully operable without the use of any Bell System-licensed UNIX software. This requirement is intended to allow for a KSOS-based system running only non-licensed software, in cases where a complete UNIX-like environment is not required.

The compiler(s) used for the development of the KSOS Kernel may be furnished by the Government.

3.1.7 Operational and Organizational Concepts

A typical KSOS installation (see Table I, below) should provide a UNIX-like environment for at least ten simultaneous, on-line users. In larger hardware configurations at least twenty simultaneous, on-line users should be supported.

3.2 Characteristics

3.2.1 Performance Characteristics

3.2.1.1 Hardware Configurations

Table I presents the minimum, typical, and maximum hardware configurations KSOS shall support. In all cases, KSOS requires:

a. A central processing unit capable of execution in at least three disjoint domains (here called kernel-, supervisor-, and user-modes). For applications of KSOS which require only the Kernel and an application program package, such as military message processing, only two disjoint domains shall be required.

b. A memory management unit which maps virtual addresses to physical memory addresses, and

c. A programmable clock with a frequency of at least 50 KHz.

KSOS shall support all hardware configurations from the minimum through the maximum. A KSOS system created for a particular hardware configuration shall be capable of operating in a reduced configuration. Operation in such reduced configurations shall not require that the KSOS system be re-created.

3.2.1.2 Software Performance

KSOS shall provide performance not less than a factor of two slower than UNIX when running Government approved benchmarks. It should be emphasized that this is an upper bound, and that the development contractor should attempt to provide performance that is as close to that of existing UNIX as possible. At least one of these benchmark tests shall include at least ten (10) terminals, including an ARPANET (logical) connection and a dial-up terminal. KSOS shall require no more than five (5) minutes to be brought up to full operation on a computer with uninitialized memory (called "rebooting"). The rebooting time does not include time for file system consistency checks or other operations performed at a given KSOS installation.

3.2.2 Physical Characteristics

This paragraph is not applicable to this specification.

3.2.3 Reliability

Because failures of the system software represent a significant potential security breach, KSOS shall be designed to be reliable and robust in the

Table I

KSOS Hardware Configurations

Item	Minimum	Typical	Maximum
Primary Memory	96 Kbytes	256 Kbytes	2 Mbytes
Secondary Memory a. Moving Head Disk (or equivalent)	4.8 Mbytes	88 Mbytes	1.4 Gbytes
b. Fixed Head Disk (or equivalent)	none	1 Mbytes	[illegible] Mbytes
Magnetic Tape (Industry compatible, nine-track)	none	1 drive	8 drives
Terminals and interfaces (96 Char ASCII)	1	10	[illegible]0
ARPANET Interface	no	no	yes
Auto. Calling Unit	0	1	1
DECtape or equiv.	0	2 drives	8 drives
Paper Tape Reader/Punch	no	no	yes
Line Printer (96 Char ASCII)	no	yes	yes

face of external (hardware) and internal failures (e.g., software inconsistencies). Where possible, KSOS shall provide graceful degradation. The effects of errors shall propagate to a minimum number of processes. Errors within a user process shall affect only that process and (possibly) other processes in the same process family.

3.2.4 Maintainability

The entire KSOS system shall be self-supporting. That is, it shall be possible to regenerate the system on itself with no outside support, i.e. no cross compilation on another system. However, the re-verification of the system's security properties may require the support of other computer systems. The organization designated to maintain KSOS should have access to

tools similar to those used in the original construction of the system.

3.2.5 Availability

This paragraph is not applicable to this specification.

3.2.6 System Effectiveness

This paragraph is not applicable to this specification.

3.2.7 Environmental Conditions

This paragraph is not applicable to this specification.

3.2.8 Nuclear Control Requirements

This paragraph is not applicable to this specification.

3.2.9 Transportability

This paragraph is not applicable to this specification.

3.3 Design and Construction

Due to the uniquely demanding requirements on KSOS, it must be designed and implemented with extraordinary precision and thoroughness. All trusted KSOS software (the Security Kernel and the trusted Non-Kernel Security Related Software) shall be formally specified with a non-procedural, mathematical language in the manner suggested by [Parnas 72]. The design shall be analyzed to demonstrate convincingly that there are no latent security flaws in it. This demonstration shall include formal proofs that the design satisfies the Government approved mathematical model of DoD security policy. These proofs shall be mechanically produced where possible.

The implementation of the trusted parts of KSOS shall be in a language amenable to formal program proofs. Representative examples of such languages include (but are not limited to) Euclid, Modula, Gypsy, and Pascal. Government approval of the language to be used is required. Provability of the code shall be an important consideration in the implementation. The computer programs shall be checked against their formal specifications utilizing a detailed checklist. Each exception condition in the specifications shall be realized as an explicit conditional statement occurring prior to the main body of the program module. The order of execution of such conditional statements in the program shall be the same as their order in the formal specification of the module.

The use of subroutines written in languages not amenable to formal proof (e.g. assembler) shall be kept to an absolute minimum. Each instance of such use shall require a written justification incorporated both as comments in the module and collected together for system wide review.

All computer programs shall include a descriptive header comment block. This shall include a functional summary of the module, its calling sequence,

its global variable usage (if any), a discussion of the algorithms used including assumptions and limitations, and a revision history. Variable declarations shall include comments describing the use of the variable. The same variable shall not be used for two unrelated functions.

3.4 Documentation

The documentation for KSOS shall consist of the following:

a. this specification

b. Type B5 (MIL-STD-483, -490), Computer Program Development Specifications for the Security Kernel, UNIX Emulator and Non-Kernel Security-Related Software.

c. Type C5 (MIL-STD-483, -490), Computer Program Product Specifications for the Security Kernel, UNIX Emulator, and Non-Kernel Security-Related Software.

d. DoD KSOS Users Manual

e. Category I Test Plans/Procedures (Computer Programming) and Category I Test Reports.

f. Category II Test Plans/Procedures (Computer Programming) and Category II Test Reports

g. System Security Plan

h. Clandestine Vulnerabilities Analysis

i. Technical Report: Security Kernel Functional Design and Interface Specifications

j. Configuration Index (Computer Programs)

k. Version Description Documents

l. Technical Report: Kernel Verification Results

m. Technical Report: Issues in KSOS Design

n. Technical Report: Final Report - KSOS Design

3.5 Logistics

3.5.1 Maintenance

Because maintenance of KSOS software may require reverification, no field maintenance of the trusted portions (Security Kernel and trusted Non-Kernel Security Related Software) of the KSOS system will be permitted. Any maintenance actions must involve the same rigor and formalism as the original design and implementation. All security related tests shall be successfully repeated

before distribution of any changes to KSOS.

3.5.2 Supply

KSOS shall normally be distributed as a nine-track, industry compatible magnetic tape. Should a particular site not possess compatible magnetic tape equipment, KSOS shall be capable of distribution on a disk pack or disk cartridge(s). A single, small supply organization should suffice to distribute KSOS to all authorized Government users.

3.5.3 Facilities and Facility Equipment

To maintain KSOS, a computer system similar to the typical configuration in Table I must be available. The software tools or their equivalents used in the design and verification of the software must also be available.

3.6 Personnel and Training

This paragraph is not applicable to this specification.

3.7 Functional Area Characteristics

3.7.1 Security Kernel

3.7.1.1 Interface

The Security Kernel shall execute in the most privileged mode(s) of the computer. The Security Kernel code shall not be modifiable while in operation. The Security Kernel shall be entered only via hardware assisted domain transitions (traps) or via hardware interrupts. The Security Kernel shall provide a secure, primitive computational environment at its interface with the next most privileged mode. The Security Kernel should provide the typical objects and corresponding manipulation primitives of Table II. (Table II is an example only. The actual objects and primitives in KSOS may differ.)

Table II

OBJECT	PRIMITIVES
Processes	Create, destroy, inter-process communication trusted invocation, timer pseudo-interrupt, get/set status
Process Segments	Create, destroy, map into/out of address space
Files	Create, remove (unlink), open, close, read, write, increment reference count (link), get/set status, get/set level, define special device file, mount/umount removable media, trusted dialog support

3.7.1.2 Performance Characteristics

The Security Kernel is a potential bottleneck for KSOS performance. High performance is an important design goal for the Security Kernel. The Security Kernel should be designed so that a minimum number of process environments are required to perform Security Kernel functions, including response to external interrupts.

3.7.1.3 Physical Characteristics

The Security Kernel should require no more than 64 Kbytes (1 Kbyte = 1024 bytes) of primary memory, including its internal tables.

3.7.2 UNIX Emulator

3.7.2.1 Interface

The UNIX Emulator shall "construct" the UNIX level objects (e.g., the hierarchical file system, the UNIX process abstraction, etc.) from the objects supported by the Security Kernel. The UNIX Emulator interface to the user-mode is described in Section 3.1.5. The UNIX Emulator shall execute in a mode (or modes) whose privilege is between that of the Security Kernel and (unprivileged) user-mode programs. The domain of execution (or address space) of the UNIX Emulator shall be disjoint from both the Security Kernel and the user program domains. Entry into the UNIX Emulator shall be via a hardware assisted domain transition (trap) from the user-mode (least privileged) of the computer. The UNIX Emulator code shall not be modifiable while in operation.

Logically, the UNIX Emulator should be part of the user process. That is, each user process should include its own (logical) copy of the UNIX Emulator. The UNIX Emulator code should be sharable to minimize the total system's memory requirements.

3.7.2.2 Performance Characteristics

The UNIX Emulator and Security Kernel should meet the performance goals of Section 3.2.1.2.

3.7.2.3 Physical Characteristics

The UNIX Emulator code and global data tables shall require no more than 64 Kbytes of primary memory. The code of the UNIX Emulator shall be sharable to minimize total system memory requirements. The per-process data tables and program stack areas should be kept as small as possible. These per-process areas should be swappable. That is, they need not be permanently resident in main memory. When the process has been swapped out (i.e. moved to secondary memory), the per-process data of the emulator should also be able to be swapped out.

3.7.3 Non-Kernel Security-Related Software

The Non-Kernel Security-Related Software is divided into four classes:

a. Secure User Services: trusted software which must have a Security Kernel supported logical path to and from it, such as system login.

b. System Operation Services: software that is necessary for the basic operation of the system, such as the Network Daemon.

c. System Maintenance Services: software that provides occasional functions needed for the continuing operation and maintenance of the system, such as file system maintenance, or the dump/restore of file systems.

d. System Administrator Services: software that facilitates the administrative control of the system, such as adding and deleting users or capturing audit information.

3.7.3.1 Interfaces

3.7.3.1.1 Secure User Services

a. Secure System Dispatcher. The Secure System Dispatcher shall provide a trusted path to a user specified trusted function, such as the File Access Modifier. The Secure System Dispatcher shall be "awakened" when a unique "escape sequence" is input from the user's terminal. The Secure System Dispatcher together with the Security Kernel shall assure that the user cannot be "spoofed" by systems pretending to be trusted functions.

b. Login. The Login mechanism shall be similar to the current login mechanism. Included in the Login mechanism in KSOS shall be the setting of the user's initial security level. Login shall be requested via the Secure System Dispatcher.

c. Logout. The Logout mechanism shall guarantee that the user is purged from the system. Logout shall be requested via the Secure System Dispatcher.

d. Newgrp. The Newgrp mechanism shall be similar to the current newgrp mechanism. It shall allow the user to change his group without doing a logout/login. Newgrp shall be requested via the Secure System Dispatcher.

e. Password Change. The Password Change mechanism shall be similar in function to the current passwd program. The development contractor should attempt to improve the present user interface. Password Change shall be requested via the Secure System Dispatcher.

f. User Level Changer. The User Security Level Changer shall allow the user to change his current level to any level less than or equal to the lower of the current system level or the user's maximum level (as specified by the System Administrator or System Security Officer). The User Level Changer shall be requested via the Secure System Dispatcher.

g. File Level Changer. The File Level Changer (here called the "regrader") shall be the mechanism for changing the level of permanent objects (files and directories). The regrader shall keep a protected log of all regrading attempts, both successful and unsuccessful. The regrader shall prevent the regrading of files to levels above the user's maximum level or to levels above the current system level. The regrader shall optionally include the "chown" (change owner) function. The regrader shall be requested via the Secure System Dispatcher.

3.7.3.1.2 System Operation Services

a. Network Daemon. The Network Daemon shall multiplex and demultiplex the data stream from the network. The Network Daemon shall be trusted to allow the support of multi-level-secure networks. The Network Daemon shall be controllable by the system operator to allow for orderly startup and shutdown, and to allow RESET'ing (or equivalent TCP commands) of foreign hosts or other similar control functions.

b. Line Printer Daemon. The Line Printer Daemon shall print previously spooled output. The Daemon shall be trusted to print the correct separators, headers and footers on the printed output. The Daemon shall also generate the appropriate records of the printing of classified material which may then be used as input to the organization's classified material accounting system if desired. The Line Printer Daemon shall also be trusted to delete files after printing.

c. Secure Mail. The Secure Mail mechanism shall be similar to the existing mail mechanism. In KSOS the Secure Mail mechanism shall assure that user's cannot read or alter the mail of others.

d. Mount/Unmount. The Mount/Unmount mechanism shall be similar to the current mechanisms. In KSOS additional steps shall be taken to improve the security and integrity of the mount mechanism. These processes shall check the suitability of the device being mounted by referring to previously generated table of acceptable mountable volumes. This table (file) shall be maintained by the Immigration Officer, discussed below.

e. System Startup and Shutdown. The System Startup mechanism shall be similar to the current startup mechanisms. The Startup Process shall include the ability to perform installation-specific actions upon system startup. The System Shutdown Process shall bring a KSOS system to an orderly shutdown including optional local actions.

3.7.3.1.3 System Maintenance Services

The System Maintenance Services shall be similar to the facilities presently provided in existing UNIX. The programs shall (of course) be adapted to the structure of the KSOS file system. The System Maintenance Services for KSOS shall include the following:

a. dump/restore of file systems

b. top-down (directory) consistency check (similar to dcheck)

c. bottom-up consistency check (similar to icheck)

d. Kernel file system manipulation program(s)

e. file system initialization (similar to mkfs)

f. system generation aids: shell files, configuration programs and checking software. This software should make the process of system generation as nearly automatic as possible.

3.7.3.1.4 System Administrator Services

The System Administrator Services software should simplify the administrative control of the system. The design goal should be to control the manipulation and updating of all system-critical files. The System Administrator Services shall include the following:

a. User Installation, Deletion and Password Overwrite. These functions should automate the process of installing and deleting users. Included in the removal of users shall be the deletion of all files owned by the user. The password overwrite function shall be an administratively controlled mechanism for handling the "forgotten password" problem. All of these function shall be restricted to be used only by the System Administrator.

b. Immigration Officer. The Immigration Officer function shall certify that a mountable file system volume is acceptable to the system. The Immigration Officer shall check the logical consistency of the volume, check that no trusted software is present, and that the user identification codes are all acceptable. The development contractor may include additional checks. The mount process shall only accept volumes previously approved by the Immigration Officer. KSOS sites should provide administrative controls on the file system volumes to assure that they are not tampered with after certification by the Immigration Officer.

c. Audit Capture. The Audit Capture Process shall capture the audit data that the Security Kernel collects and shall copy it into a file for later reduction. The development contractor is not expected to provide this audit reduction software.

d. Security Officer Support. The development contractor shall provide additional functions needed to support the System Security Officer. Government approval of any proposed software in this area is required.

3.7.3.2 Performance Characteristics

The performance requirements of the Non-Kernel Security Related Software vary by the different groups of this software. The System Operation Services can be thought of as extensions of the Security Kernel and Emulator. Thus, this software must be designed and implemented to offer high performance. The other three groups, Secure User, System Maintenance and System Administrator

Services are not very demanding in their performance requirements.

3.7.3.3 Physical Characteristics

The amount of dedicated memory consumed by the System Operation Services software should be minimized. The number of distinct process address spaces utilized by each of the Non-Kernel Security Related Software functions should be minimized.

3.8 Precedence

The primary requirement of KSOS is provable security. Design trade-offs between security and other factors shall be resolved in favor of security. The secondary requirements are performance and compatibility. Both are relatively equal in importance. Other less critical requirements are clarity and maintainability.

In the event of an inconsistency between this specification and the referenced documents (Section 2), the inconsistency shall be resolved by reference to the documents in the following order:

a. the contractual provisions of the development contract

b. this (Type A) System Specification

c. the Type B5 Development Specifications

d. the Type C5 Product Specifications

4. QUALITY ASSURANCE PROVISIONS

4.1 General

The underlying philosophy of KSOS testing is to provide a convincing demonstration of the security and completeness of the KSOS system. Testing and quality assurance are a complement to the formal verification aspects of KSOS. All testing on KSOS shall be conducted at the development contractor's facility.

KSOS shall be subject to the technical reviews and audit procedures of MIL-STD-1521A, Appendices A-F. The following technical reviews and audits will be conducted:

a. System Requirements Review (Appendix A)

b. System Design Review (Appendix B)

c. Preliminary Design Review (Appendix C)

d. Critical Design Review (Appendix D)

e. Functional Configuration Audit (Appendix E)

f. Physical Configuration Audit (Appendix F)

To provide data for assessing KSOS reliability, all system reloads (rebootings) during formal testing shall be noted in the test reports. File system consistency shall be checked before and after all formal tests, with the results noted in the test reports. The Category II testing shall include at least one test of long duration, at least twenty-four (24) hours. This long duration test shall include Government approved scenarios which are representative of typical KSOS intended uses.

To ensure completeness and repeatability as much testing as practical shall be automated.

The Critical Design Review shall include a review of any mathematical proofs showing that the design meets the requirements of the Government approved DoD security model.

The Functional Configuration Audit shall be the vehicle for the formal review of the design versus the mathematical description. The Physical Configuration Audit shall be the vehicle for the formal review of the "as built" computer programs versus both their design and their supporting documentation. Both audits shall include review of any relevant mathematical proofs of correctness.

4.1.1 Responsibility for Tests

The development contractor shall conduct all KSOS testing from Government approved test plans and procedures. Test reports shall be prepared for all formal tests and submitted to the Government for approval. The Government

may elect to witness selected tests. All testing shall utilize Government approved quality assurance procedures.

4.2 Quality Conformance Inspections

The Category I and II tests (Section 4.1) and the technical reviews and audits (MIL-STD-1521A, Appendices A-F and Section 4.1) shall be the mechanisms by which compliance with the requirements of Section 3. is determined. Table III shows how each of the requirements from Section 3. will be shown to have been met.

Table III

Requirement	Paragraph	How*	Test Type
UNIX Compatible Interface	3.1	I,D	Cat I,Cat II
System Security Properties	3.1.1	I,D,F	Cat II, Formal Verification Results
Threats	3.1.1	I,D,F	Cat II, Formal Verification Results
System Internal Structure	3.1.4	I	Cat II
UNIX compatability	3.1.5	D	Cat I,Cat II
Hierarchical File System	3.1.5	I	Cat I
User-mode Kernel Calls	3.1.5	D	Cat I,Cat II
UNIX Call Interface	3.1.5.1	I,D	Cat I,Cat II
UNIX Calls	3.1.5.1.1 thru 3.1.5.1.43, 3.1.5.2.1 thru 3.1.4.2.4, 3.1.5.3 thru 3.1.5.7	I,D	Cat I, Cat II
UNIX compatability	3.1.6	D	Cat II
No Bell-licensed code	3.1.6	I	Unit test,Cat I
Number of users	3.1.7	D	Cat II
Hardware configurations	3.2.1.1	I,D	Cat II
Software Performance	3.2.1.2	D	Cat II
Reliability	3.2.3	I,D	Cat II
Maintainability	3.2.4	I,D	Cat II
Design and construction	3.3	I,F	Unit Test,Cat I, Cat II, Formal Verification Results

Security Kernel Interface	3.7.1.1	I,F	Cat I,Cat II Formal Verification Results
Kernel Performance	3.7.1.2	I,D	Cat I,Cat II
Kernel Size	3.7.1.3	I	Cat I
Emulator Interface	3.7.2.1	I,D	Cat I
Emulator Performance	3.7.2.2	I,D	Cat II
Emulator Size	3.7.2.3	I	Cat I
NKSR Interfaces	3.7.3.1.1 thru 3.7.3.1.4	I,D	Cat I,Cat II
NKSR Performance	3.7.3.2	D	Cat II
NKSR Sizes	3.7.3.3	I	Cat I

*D Demonstration
I Inspection
F Formal Verification

5. PREPARATION FOR DELIVERY

This section is not applicable to this specification.

6. NOTES

This section is not applicable to this specification.

10. APPENDIX

This appendix at the time of this writing reflects the difference ι the KSOS system as built and the KSOS system as specified in the type A system specification. No justification is given for the differences listed here.

10.1 Hardware Configuration

The amount of memory required to support a running KSOS system has increased. The current memory requirements are:

Primary Memory	Minimum	Typical	Maximum
	256 Kbytes	750 Kbytes	2 Mbytes

The auto calling unit was by customer direction deleted from the list of KSOS supported devices.

10.2 Software Performance

KSOS system performance is slower than projected. KSOS was to provide performance not less than a factor of two slower than UNIX when running Government approved benchmarks. Though no formal timings have been done to date, current estimates show that KSOS is approximately 6 or 7 times slower than UNIX.

10.3 Maintainability

It was desired that it would be possible generate a new KSOS system on a KSOS system. This type of generation has not been attempted. Since KSOS was developed on a PWB/UNIX system and KSOS provides a UNIX 6.0 environment, some amount of work would have to be done transfer the KSOS generation programs to be compatible with UNIX 6.0.

10.4 Design and Construction

It was desired that all trusted KSOS software be formally specified. However upon agreement between the customer and this contractor the following list of Non-Kernel Security Related software was not specified:

ACP - Audit Capture
DCC - Directory Consistency Checker
DPE - Device Profile Editor
FSD - File System Dump
FSI - File System Initialization
FSR - File System Resor‹
IOP - Immigration Officer Process
KPN - Kernel to Path Name Mapping
KSG - KSOS System Generation
LPC - Level Preserving Copy
LPP - Level Preserving Print
MCE - Modify Control Entry
PCE - Privilege Control Editor
SME - Security Map Editor
SMP - Secure Mail Process
SPE - System Profile Editor
STC - Storage Consistency
SSU - System Start Up
TPE - Terminal Profile Editor
UCE - User Control Editor

It was desired that all the the trusted software for KSOS be written in a language amenable to formal proofs. This was not done for all Non-Kernel Security Related software. Upon agreement between the customer and this contractor the following NKSR software was written in C:

LPD - Line Printer Demon
UCE - User Control Editor
TPE - Terminal Profile Editor
DPE - Device Profile Editor
SME - Security Map Editor
SPE - System Profile Editor

10.5 Kernel and Emulator Physical Characteristics

It was desired that the kernel and the emulator should require no more than 64 Kbytes each. However both currently require 128 Kbytes.

FILMED

4-8